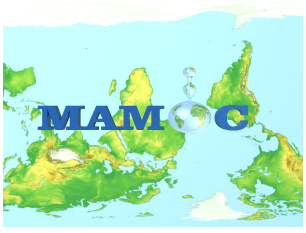
HPMviz:

Simulation, analysis and visualization of Hamiltonian Particle-Mesh method

version 1.6, January 18, 2012

Vladimir Molchanov, Lars Linsen



Contents

Introduction 3

1 General information 3
1.1 License 3
1.2 Data format 4

2 User interface 7

3 Tutorial 7

4 What to expect next 7

A The Burgers Solution to SWE 8

B The Cosine Vortex Solution to SWE 9

References 10

Introduction

Hamiltonian Particle-Mesh method (HPM) method was proposed by J. Frank, G. Gottwald and S. Reich [1, 2, 3]. The authors modified ideas of Smoothed Particle Hydrodynamics (SPH) and Particle-in-Cell algorithms. The new Lagrangian-Eulerian method was successfully applied for numerical solution of Shallow Water Equations (SWE), which are extensively used to simulate geophysical processes. HPM is a hybrid numerical method. It has a natural adaptivity property like most particle methods. At the same time it incorporates the procedure of artificial smoothing of the depth field over a regular grid into the classical Lagrangian formalism. Usually the inverse Helmholtz operator is applied. This smoothing reduces the numerical noise and can be viewed as an efficient implementation of SPH for globally supported basis functions.

Many recent studies have focused on robustness, accuracy and conservation properties of the HPM method. It was shown that HPM equations inherit the Hamiltonian structure of SWE, advect the potential vorticity properly and have better long-time stability properties than SPH. This was also tested in a number of numerical experiments [4]. However, little research has been done in the field of optimization of the method and analytical investigation of its convergence.

There are three characteristic scales for the system of N interacting particles in d-dimensional space, whose evolution is governed by HPM equations. They are: The artificial smoothing length μ, the average particle distance l_{part} and the mesh size λ. The interrelation of the scales is a subject of recent study [5].

HPMviz is a tool which serves to perform simulations using HPM method, investigate the behavior of the solution with respect to the key parameters (μ, l_{part}, λ, type of the kernel functions, etc.), and visualize the results. It has been (and is still being) developed at Visualization and Computer Graphics Laboratory (VCGL), Jacobs University, Bremen, Germany (http://vcgl.jacobs-university.de).

1 General information

The code is written in C++. Fourier transformations are performed using FFTW library. We used OpenGL for 3D visualization and GLSL for the Phong shading. The GUI was developed in Qt 4.7.

1.1 License

This program is free software; you can redistribute it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

1.2 Data format

The user can choose one of built-in flows to perform a simulation. These flows are known exact solutions of the (Rotating) SWE: the (boring :) uniform flow, the Burgers solution, the cosine vortex steady flow. See Appendix A and B for description and discussion. Alternatively, one can load his/her own data about initial depth profile, topology values and velocity field sampled on a 2D uniform grid. These data will be then interpolated (two-linear scheme) to produce required fields sampled on particles. The depth, topology and velocity fields should be written into *filename.depth*, *filename.top* and *filename.vel* binary files, correspondingly. The format of files with scalar data is as follows:

```
Writing depth/topology data file:
begin
  write(nx);                          //number of grid nodes in x-direction, unsigned long;
  write(ny);                          //number of grid nodes in y-direction, unsigned long;
                                                  //scalar field, nx×ny×double;
  for i := 1 to nx do
     for j := 1 to ny do
        write(f[i][j]); od od
end
```

The format of file with vector data is as follows:

```
Writing velocity data file:
begin
  write(nx);                          //number of grid nodes in x-direction, unsigned long;
  write(ny);                          //number of grid nodes in y-direction, unsigned long;
                                                  //vector field, nx×ny×3×double;
  for i := 1 to nx do
     for j := 1 to ny do
        write(f[i][j]); write(v[i][j].y); write(v[i][j].z); od od
end
```

To make your and our life easier, we implemented a functions which write all parameters into a .prm file / read parameters from a .prm file. An example of such a file is put into **/data** folder in the project. The file can be modified by hand, if needed, since it is in a text format. To load the recorded parameters either append the file name when launching the program (./hpmviz filename.prm) or choose choose **Load parameters** in **Menu bar** → **File**. The description of parameter file follows:

-run = ⟨ (0 or 1) if 1, immediate launch of simulation after reading parameters ⟩

-run_series = ⟨ (0 or 1) if 1, immediate launch of series of simulations after reading parameters ⟩

-qkin = ⟨ (0 or 1) if 1, compute kinetic error ⟩

-qkin = ⟨ (0 or 1) if 1, compute kinetic error ⟩

-qpot = ⟨ (0 or 1) if 1, compute potential error ⟩

-qtot = ⟨ (0 or 1) if 1, compute total error ⟩

-ekin = ⟨ (0 or 1) if 1, compute kinetic energy ⟩

-epot = ⟨ (0 or 1) if 1, compute potential energy ⟩

-etot = ⟨ (0 or 1) if 1, compute total energy ⟩

-maxqkin = ⟨ (0 or 1) if 1, compute maximal kinetic error ⟩

-maxqpot = ⟨ (0 or 1) if 1, compute maximal potential error ⟩

-maxqtot = ⟨ (0 or 1) if 1, compute maximal total error ⟩

-lmomentum = ⟨ (0 or 1) if 1, compute linear momentum ⟩

-apd = ⟨ (0 or 1) if 1, miltiply the smoothing length with average particle distance in x-direction ⟩

-cnppc = ⟨ (0 or 1) if 1, keep constant number of particles in cell during multiple runs ⟩

-gx = ⟨ (unsigned long) number of grid nodes in x-direction ⟩

-gy = ⟨ (unsigned long) number of grid nodes in y-direction ⟩

-nx = ⟨ (unsigned long) number of particles in x-direction ⟩

-ny = ⟨ (unsigned long) number of particles in y-direction ⟩

-invHelm = ⟨ (unsigned long) power of the inverse Helmholtz operator (global smoothing) ⟩

-mu = ⟨ (non-negative double) global smoothing length ⟩

-dt = ⟨ (non-negative double) time step ⟩

-T = ⟨ (non-negative double) final time ⟩

-g = ⟨ (double) gravity ⟩

-f = ⟨ (double) Coriolis parameter ⟩

-eps = ⟨ (double) error threshold for the Burgers solution ⟩

-ic = ⟨ (0, …, 3) initial conditions choice ⟩

-lc = ⟨ (0, …, 5) local kernel choice ⟩

-mc = ⟨ point-wise (0) or trapezoidal (1) evaluation of particle mass ⟩

-im = ⟨ (0, …, 2) integration method choice ⟩

-dt_save = ⟨ (double) time step for data recording ⟩

-xres = ⟨ (unsigned long) resolution in x-direction ⟩

-yres = ⟨ (unsigned long) resolution in y-direction ⟩

-save1 = ⟨ (int) 1st parameter in multi-run ⟩

-save2 = ⟨ (int) 2nd parameter in multi-run ⟩

-delta1 = ⟨ (double) increment of 1st parameter ⟩

-delta2 = ⟨ (double) increment of 2nd parameter ⟩

-loop1 = ⟨ (double) steps for 1st parameter ⟩

-loop2 = ⟨ (double) steps for 2nd parameter ⟩

-lin1 = ⟨ exponential (0) or linear (1) change of the 1st parameter ⟩

-lin2 = ⟨ exponential (0) or linear (1) change of the 1st parameter ⟩

-name = ⟨ (string) name of experiment ⟩

Value of $-run$ is ignored, if $-run_series = 1$. Initial conditions: $-ic = 0$ constant depth, $-ic = 1$ the 1D Burgers solution, $-ic = 2$ the 2D cosine vortex steady solution, $-ic = 3$ step-function. Local kernel: $-lc = 0$ quadratic spline, $-lc = 1$ cubic spline, $-lc = 2$ quartic spline, $-lc = 3$ quintic spline, $-lc = 4$ constant, $-lc = 5$ linear spline (splines will be re-ordered in the next version). Integration method: $-im = 0$ Euler 1st-order, $-im = 1$ Runge-Kutta 2nd-order, $-im = 2$ Runge-Kutta 4th-order. Parameters for series of runs: $-save1 = 0$ global smoothing, $-save1 = 1$ number of grid nodes in x-direction, $-save1 = 2$ number of grid nodes in y-direction, $-save1 = 3$ number of grid nodes in both x- and y-direction, $-save1 = 4$ number of particles in x-direction, $-save1 = 5$ number of particles in y-direction, $-save1 = 6$ number of particles in both x- and y-direction, $-save1 = 7$ power of the inverse Helmholtz operator, $-save1 = 8$ local spline. The same applies to $-save2$.

Example of parameters setting for series of runs: $-save1 = 0$, $-delta1 = 0.05$, $-loop1 = 3$, $-lin1 = 1$, $-mu = 0.0$ result in the global smoothing parameter change as follows: $\mu = 0.0, 0.05, 0.1$. Another example of parameters setting: $-save2 = 1$, $-delta2 = 2$, $-loop2 = 4$, $-lin2 = 0$, $-gx = 100$ result in the number of the grid nodes in x-direction change as follows: $gx = 100, 200, 400, 800$. If $-cnppc = 1$ and $-nx = 200$, the number of particles in x-direction will be also changed as follows: $nx = 200, 400, 800, 1600$. If $-cnppc = 0$, nx will remain unchanged.

2 User interface

Main widgets of the user interface are:

Menu bar on top of the main window. It includes read/write options and contron on the light source parameters. Short information about the program and authors can be called their.

GL window in the center of the main window. 3D view of the simulated flow and 2D plots of the computed quantities can be displayed there.

Slider to the left from the GL window. Allows the user to change the scale of the 3D view in z-direction.

Tab widget to the right from the GL window. **Show** tab has a number of checkBoxes to enable/disable elements on the 3D view, to change their colors and to choose a proper shader; **Functionals** tab has a set of checkBoxes to mark the quantities which should be computed during simulation, recorded afterwards, and shown in a new GL window after the user presses **Show results** button; **Series** tab includes widgets to set parameters for a series of runs.

Lower panel allows to change a lot of parameters of the current simulation including name of the test, gravity value, Coriolis parameter, grid resolution, number of particles, smoothing length, exponent of the smoothing operator, time integration method, time step, final time, etc. **Recorded data** window called when clicking on **Tab widget** → **Functionals** → **Show results** button contains a number of checkBoxes to enable/disable plots of the recorded data and change their colors and thickness of lines. Data can be also read from a file by clicking on **Read from file** button.

3 Tutorial

Run the program, choose **Load parameters** in **Menu bar** → **File**, choose a *.prm file to open. Alternatively, you can set up all parameters on the **Lower panel** and then save them clicking on **Menu bar** → **File** → **Save parameters**.

When **avr. part. dist. x** checkBox is checked, the global smoothing parameter μ will be multiplied with the average particle distance in x-direction, i.e., with $2\Pi/\mathrm{nx}$ value.

If **Record step** lineEdit value differs from the value specified in **Time step** lineEdit, the computed quantities (errors, energies, ...) will be resampled using linear interpolation.

Click **Run** button to launch a single run or **Run series** to perform a series of runs according to parameters chosen on **Tab widget** → **Series** panel.

A screenshot of the 3D view can be saved as a *.bmp file when triggering **Menu bar** → **File** → **Screenshot**.

4 What to expect next

Currently, we gather feedbacks from interested people, perform our research on optimization of the HPM method, and look for programming bugs in the code. As soon as a relatively

stable state will be achieved, the following features are to be available:

- Better manual :)
- Streamlines and pathlines tracking
- Rendering of potential vorticity field
- More nice built-in transfer functions and colors, textures
- Parallel implementation using OpenMP
- The feature you need the most (provided you send us your feedback)

In the current version we have disabled the functions writing the computed errors into files. This option will be enabled as soon as we complete our study on optimization of HPM.

A The Burgers Solution to SWE

As discussed in [7], the one dimensional SWE with vanishing topography

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = -g\frac{\partial h(x,t)}{\partial x}, \tag{1}$$

$$\frac{\partial h(x,t)}{\partial t} + \frac{\partial (u(x,t)h(x,t))}{\partial x} = 0, \tag{2}$$

can be reduced to the inviscid Burgers equation

$$\frac{\partial q(x,t)}{\partial t} + q(x,t)\frac{\partial q(x,t)}{\partial x} = 0, \tag{3}$$

provided $u + 2\sqrt{gh} = c$, $q(x;\, t) = c - 3\sqrt{gh}$, c is a constant. Note, however, that the correct form of equation (30) in [7] reads

$$h(x,t) = \frac{(q(x,t)-c)^2}{9g}, \qquad u(x,t) = \frac{2q(x,t)+c}{3}. \tag{4}$$

Substitution h and u into (1) and (2) results in

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} + g\frac{\partial h(x,t)}{\partial x} =$$

$$\frac{2}{3}\frac{\partial q(x,t)}{\partial t} + \frac{2q(x,t)+c}{3}\cdot\frac{2}{3}\frac{\partial q(x,t)}{\partial x} + \frac{2(q(x,t)-c)}{9}\frac{\partial q(x,t)}{\partial x} =$$

$$\frac{2}{3}\frac{\partial q(x,t)}{\partial t} + \frac{2}{3}q(x,t)\frac{\partial q(x,t)}{\partial t} = 0$$

and

$$\frac{\partial h(x,t)}{\partial t} + u(x,t)\frac{\partial h(x,t)}{\partial x} + h(x,t)\frac{\partial u(x,t)}{\partial x} =$$

$$\frac{2(q(x,t)-c)}{9g}\frac{\partial q(x,t)}{\partial t} + \frac{(q(x,t)-c)^2}{9g}\cdot\frac{2}{3}\frac{\partial q(x,t)}{\partial x} + \frac{2q(x,t)+c}{3}\cdot\frac{2(q(x,t)-c)}{9g}\frac{\partial q(x,t)}{\partial x} =$$

$$\frac{2(q(x,t)-c)}{9g}\left[\frac{\partial q(x,t)}{\partial t} + \left(\frac{q(x,t)-c}{3} + \frac{2q(x,t)+c}{3}\right)\frac{\partial q(x,t)}{\partial x}\right] = 0,$$

correspondingly, due to equation (3).

The Cauchy problem for equation (3)

$$\frac{\partial q(x,t)}{\partial t} + q(x,t)\frac{\partial q(x,t)}{\partial x} = 0, \tag{5}$$

$$q(x,0) = q_0(x), \tag{6}$$

can be solved by characteristics method. First, we find integrals of the system

$$\frac{\mathrm{d}t}{1} = \frac{\mathrm{d}x}{q} = \frac{\mathrm{d}q}{0}.$$

Clearly, $q = C_1$ is the first integral. Then, $q\mathrm{d}t = \mathrm{d}x$ can be integrated producing $qt - x = C_2$. So, the general solution has the form $F(q, qt - x) = 0$. To satisfy the initial conditions, constants C_1 and C_2 should be defined. A parametrization of the initial condition reads

$$t = 0, \qquad x = \xi, \qquad q = q_0(\xi).$$

Then, C_1 and C_2 can be found from the system

$$C_1 = q_0(\xi), \qquad C_2 = -\xi.$$

Finally, the solution of the Cauchy problem has the form

$$q(\xi, 0) = q_0(\xi), \qquad x = \xi + q_0(\xi)t.$$

B The Cosine Vortex Solution to SWE

SWE in two spatial dimensions with topography $b(x,y)$ and Coriolis parameter f read

$$\frac{\partial u(x,y,t)}{\partial t} + u(x,y,t)\cdot\nabla u(x,y,t) + f u^{(}\perp)(x,y,t) = -g\nabla(h(x,y,t) + b(x,y)), \tag{7}$$

$$\frac{\partial h(x,y,t)}{\partial t} + u(x,y,t)\cdot\nabla h(x,y,t) = -h(x,y,t)\nabla\cdot u(x,y,t), \tag{8}$$

where $u = (u,v)$ and $u^{(}\perp) = (-v,u)$.

A cosine vortex solution (see [6, 7]) considered in the code is given by the following expressions:

$$h(x,y) = 2.5 - (\cos x + \cos y)/g, \tag{9}$$

$$b(x,y) = 1.0 - f\cos x\cos y/g, u(x,y) = (-\sin y, \sin x). \tag{10}$$

This solution is steady, i.e., it does not depend on time t. So, we conclude $u_t \equiv h_t \equiv 0$. Then,

$$u^{\perp} = (-\sin x, -\sin y),$$

$$u \cdot \nabla u = (uu_x + vu_y, uv_x + vv_y) = (-\sin x \cos y, -\sin y \cos y),$$
$$g\nabla(h+b) = (-f\sin x - \cos y \sin x, -f\sin y - \cos x \sin y),$$
$$\nabla h = (\sin x, \sin y),$$
$$u \cdot \nabla h = -\sin x \sin y + \sin y \sin x = 0,$$

finally,

$$\nabla \cdot u = 0.$$

Inserting all expressions into (7), we complete the proof.

References

1. J. Frank, G. Gottwald, S. Reich. A Hamiltonian Particle-Mesh Method for the Rotating Shallow Water Equations. *In M. Griebel and M.A. Schweitzer, eds., Meshfree Methods for Partial Differential Equations, Lecture Notes in Computational Science and Engineering, Springer*. 26:131-142, 2002.

2. J. Frank, S. Reich. The Hamiltonian Particle-Mesh Method for the Spherical Shallow Water Equations. *Atmos. Sci. Let.* 5:89-95, 2004.

3. C. J. Cotter, J. Frank, S. Reich. Hamiltonian Particle-Mesh Method for Two-Layer Shallow-Water Equations Subject to the Rigid-Lid Approximation. *SIAM J. Applied Dynamical Systems*. 3:69-83, 2004.

4. J. Frank, S. Reich. Conservation Properties of Smoothed Particle Hydrodynamics Applied to the Shallow Water Equations. *BIT*. 43:40–54, 2003.

5. V. Molchanov. Particle-Mesh and Meshless Methods for a Class of Barotropic Fluids. *PhD Thesis, School of Engineering and Science, Jacobs University Bremen*, 2008.

6. R. Iacono. Analytic Solutions to the Shallow Water Equations. *Phys. Rev. E*. 72, 2005.

7. V. R. Ambati, O. Bokhove. Space-time discontinuous Galerkin discretization of rotating shallow water equations. *J. Comp. Phys.* 225:1233-1261, 2007.

8. P. Tassi, O. Bokhove, C. Vionnet. Space-discontinuous Galerkin method for shallow water ows kinetic and HLLC ux, and potential vorticity generation. *Adv. Water Res.* 30:998-1015 2007.